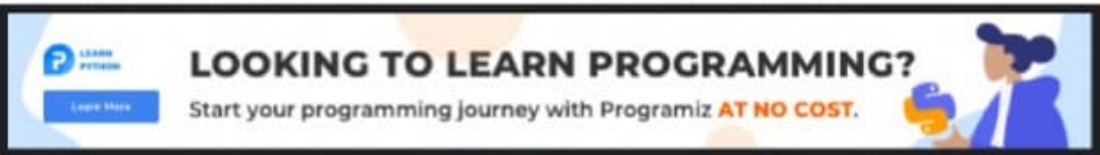
Online C Compiler - Programiz | You are given with an array an | ChatGPT

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



Programiz PRO ›

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>

typedef enum { RED, BLACK } Color;

typedef struct Node {
    int data;
    Color color;
    struct Node* left;
    struct Node* right;
    struct Node* parent;
} Node;

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->color = RED;
    newNode->left = newNode->right = newNode->parent = NULL;
    return newNode;
}

void leftRotate(Node** root, Node* x) {
    Node* y = x->right;
    x->right = y->left;
    if (y->left != NULL) {
        y->left->parent = x;
```

```
/tmp/1wCRJ5OFSq.o
Red-Black Tree:
        14 (RED)
    13 (BLACK)
        10 (RED)
8 (BLACK)
            7 (RED)
        6 (BLACK)
            4 (RED)
    3 (RED)
        1 (BLACK)


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 10:40 26-08-2024

Online C Compiler - Programiz | You are given with an array arr | ChatGPT

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

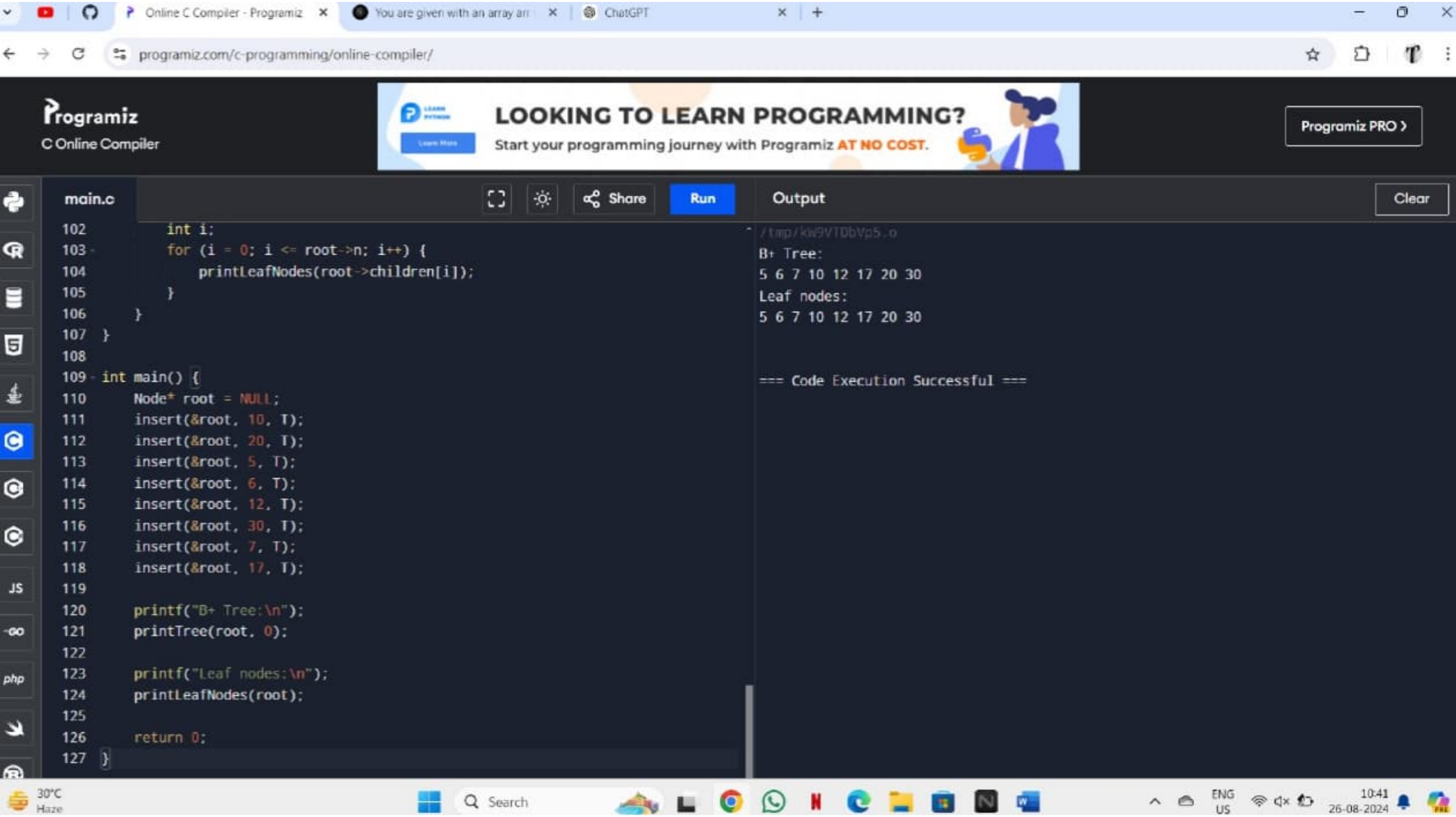
Programiz PRO >

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>

#define T 3

typedef struct Node {
    int *keys;
    struct Node **children;
    int n;
    int leaf;
} Node;

Node* createNode(int leaf) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->keys = (int*)malloc((2*T-1)*sizeof(int));
    newNode->children = (Node**)malloc((2*T)*sizeof(Node*));
    newNode->n = 0;
    newNode->leaf = leaf;
    return newNode;
}

void insert(Node** root, int k, int t) {
    Node* r = *root;
    if (r == NULL) {
        *root = createNode(1);
        (*root)->keys[0] = k;
```

```
/tmp/z3IP21vA75.o
B-Tree:
5 6 7 10 12 17 20 30


=== Code Execution Successful ===
```

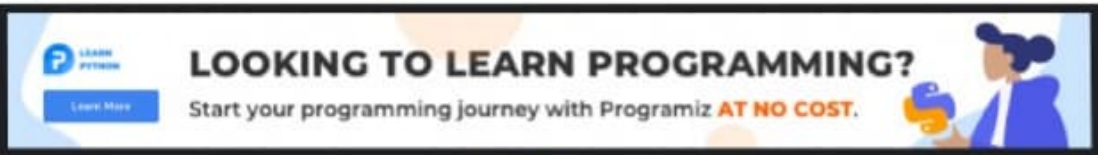
30°C Haze | Search | ENG US | 10:40 26-08-2024


Online C Compiler - Programiz
You are given with an array arr
ChatGPT
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
LOOKING TO LEARN PROGRAMMING?
Start your programming journey with Programiz AT NO COST.
Programiz PRO
main.c
Share
Run
Output
Clear
int i;
for (i = 0; i <= root->n; i++) {
printLeafNodes(root->children[i]);
}
}
}
int main() {
Node* root = NULL;
insert(&root, 10, T);
insert(&root, 20, T);
insert(&root, 5, T);
insert(&root, 6, T);
insert(&root, 12, T);
insert(&root, 30, T);
insert(&root, 7, T);
insert(&root, 17, T);
printf("B+ Tree:\n");
printTree(root, 0);
printf("Leaf nodes:\n");
printLeafNodes(root);
return 0;
}
/tmp/kW9VTDbVp5.o
B+ Tree:
5 6 7 10 12 17 20 30
Leaf nodes:
5 6 7 10 12 17 20 30
=== Code Execution Successful ===
30°C
Haze
Search
ENG
US
10:41
26-08-2024

Online C Compiler - Programiz | You are given with an array arr | ChatGPT

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



Programiz PRO >

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
    int height;
} Node;

int height(Node* node) {
    if (node == NULL) {
        return 0;
    }
    return node->height;
}

int max(int a, int b) {
    return (a > b) ? a : b;
}

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    newNode->height = 1;  // New node is initially added at leaf
```

Output | Clear

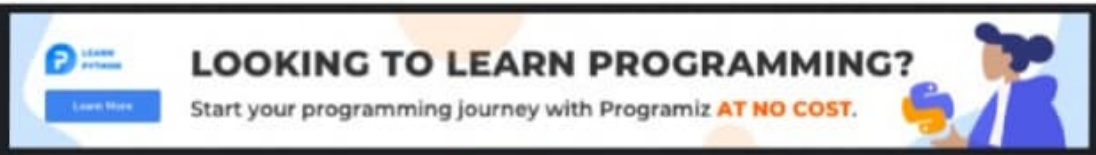
```
/tmp/GjS0WvptbN.o
Preorder traversal of the constructed AVL tree is:
30 20 10 25 40 50
Preorder traversal after deletion of 40:
30 20 10 25 50
Node with data 25 found in the tree.


=== Code Execution Successful ===
```

30°C | Search | ENG | 10:38

Online C Compiler - Programiz | You are given with an array arr

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler



Programiz PRO >

main.c | Share | Run | Output | Clear

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

Node* insert(Node* root, int data) {
    if (root == NULL) {
        root = createNode(data);
    } else if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}
```

```
/tmp/OSiR5I7eyO.o
Found: 40
Min: 20
Max: 80


=== Code Execution Successful ===
```

30°C Haze | Search | ENG US | 10:36 26-08-2024